INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 11(2):167-178, April 2016.

Improved Performance by Combining Web Pre-Fetching Using Clustering with Web Caching Based on SVM Learning Method

K.R. Baskaran, C. Kalaiarasan

Kuttuva Rajendran Baskaran*
Department of Information Technology
Kumaraguru College of Technology, Coimbatore, India
*Corresponding author: krbaski@yahoo.com

Chellan Kalaiarasan
Tamilnadu College of Engineering, Coimbatore, India
ckalai2001@yahoo.com

Abstract: Combining Web caching and Web pre-fetching results in improving the bandwidth utilization, reducing the load on the origin server and reducing the delay incurred in accessing information. Web pre-fetching is the process of fetching the Web objects from the origin server which has more likelihood of being used in future. The fetched contents are stored in the cache. Web caching is the process of storing the popular objects "closer" to the user so that they can be retrieved faster. In the literature many interesting works have been carried out separately for Web caching and Web pre-fetching. In this work, clustering technique is used for pre-fetching and SVM-LRU technique for Web caching and the performance is measured in terms of Hit Ratio (HR) and Byte Hit Ratio (BHR). With the help of real data, it is demonstrated that the above approach is superior to the method of combining clustering based pre-fetching technique with traditional LRU page replacement method for Web caching.
Keywords: Classification, support, confidence, hit ratio, byte hit ratio, web pre-fetching, web caching.

1 Introduction

In recent times, with rapid growth of WWW, there is ever increasing demand for computer networking resources. With increased number of Web based applications created by many users, the increase in bandwidth does not address the delay problems [5]. The existing prediction algorithms often predict relevant as well as irrelevant pages. In caching the pre-fetched Web pages, efficient cache replacement techniques have to be deployed to manage the cache content. The traditional cache replacement techniques used does not increase the cache hit ratio to a great extent. Machine learning techniques are deployed to improve the performance of Web proxy caching methods [2]. Compared to traditional caching approaches, intelligent Web caching methods are more efficient. Details about intelligent caching methods are found in [1]. The remaining parts of this paper are organized as follows. Section 2 gives an overview of Web caching and Web Pre-fetching techniques. Section 3 contains the proposed block diagram (with description) and the steps involved in the proposed method of combined clustering (intra clustering also considered) based pre-fetching technique with machine learning technique (SVM algorithm). It also contains the algorithm for the combined intelligent Caching (SVM) and Pre-fetching. Section 4 discusses about performance evaluation and section 5 concludes the paper and suggests possible future works.

2 Web caching and Web Pre-fetching

Enhancing the performance of Web based systems is possible by Web caching in which, the Web objects which have high probability of being accessed in the near future are kept closer to the user either in the client's machine or in the proxy server.

The factors (features) of Web objects that influence Web proxy caching considered in this work are, namely: recency (object's last reference time), frequency (number of requests made to an object), size (size of the requested Web object) and access latency of Web object.

The standard metrics used to analyze the performance of web caching methods are Hit Ratio (HR) and Byte Hit Ratio (BHR). HR is the percentage of number of requests that are served by the cache over the total number of requests.

BHR is the percentage of the number of bytes that correspond to the requests served by the cache over the total number of bytes requested. A high HR indicates the presence of the requested object in the cache most of the time and high BHR indicates reduced user-perceived latency and savings in bandwidth.

3 The proposed method of combined clustering based pre-fetching technique with machine learning technique

As shown in the Fig. 1, Web pages accessed by various users are identified from the log file. Web log file contents are preprocessed and trained using the features namely: recency, frequency, retrieval time and size of web object. Dataset is created from the proxy log file. Web navigation graph (WNG) is constructed for each user using a user's session time interval of thirty minutes. It is considered that the two subsequent requests should not have a time interval greater than thirty minutes. At the end of each time interval new navigation graph is constructed for each user based on the log files contents. WNGs show the navigations made by various users between various Web objects for inter-site clustering and between various Web pages with in a Web object for intra-site clustering. Each node in the WNG represents a Web object requested by the user and each edge represents user's transitions from one Web object to another and a weight is assigned to each edge which represents the number of transitions between those nodes. A clustering algorithm gets the contents of WNGs as inputs and two parameters namely Support and Confidence are used to keep track of frequently visited objects/pages by the user. By fixing a threshold value for these parameters, edges which have values less than this threshold can be removed [4]. Support is defined as the frequency of navigation between two nodes u1 and u2. The confidence is defined as freq(u1,u2)/pop(u1) where pop(u1) is the popularity of u1. Popularity of a node is the number of incoming edges in to that node. The WNG is partitioned in to sub graphs by removing those edges that have low Support and Confidence values. The nodes in each connected sub-graph become a cluster. When a user requests any one of the nodes in a cluster and if it is found in the long- term cache or in the short-term cache, all the remaining nodes in that cluster along with all the intra pages of the requested node can be pre-fetched in to the short-term cache if it is not in the short- term cache or in the long-term cache [4].

This pre-fetching is done during the browser idle time and this pre-fetching helps in reducing the user perceived latency time. If a Web object is accessed by the user for an access count greater than the threshold then that Web object are moved from the short term cache to long term cache. LRU method is used for removal of objects from the short term cache if sufficient space is not available for caching a new Web object.

When the objects are moved from short term cache to long term cache SVM classifier is used to classify the Web objects as class 0 or class 1 [2]. When a request is made for a Web object,

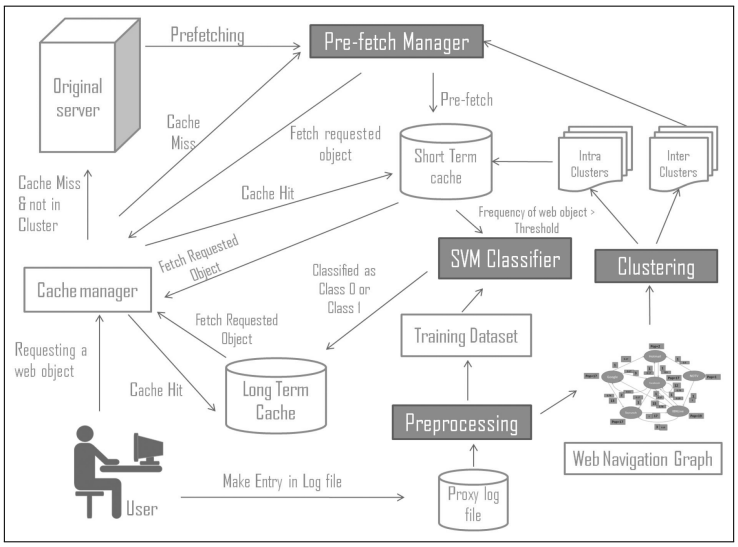

Figure 1: Block diagram

simultaneous search is made in short term cache as well as in long term cache. If that Web object is available in the short-term cache, its access count is increased by one. When the access count becomes greater than the threshold value then that Web object is given as input to SVM classifier for classification. If the classifier classifies it as class 0, it is moved to the bottom of long-term cache else if classified as class 1 then it is moved to the top of long-term cache. If sufficient space is not found in the long-term cache, then the Web object placed at the bottom of the long-term cache are removed to offer space for the new Web object. At the same time a copy of this Web object is transmitted to the requested user. If this object is found in the long-term cache, re-classification of that object is performed.

If it is re-classified as class 1, it is moved to the top of the cache (long-term cache) else it is moved to the bottom of that cache. It is then transmitted to the requested user and pre-fetching of other Web objects if any that belong to that cluster as well as all the intra pages of this Web object in to the short-term cache is initiated. If cache miss occurs then the requested object is searched in all the clusters generated by clustering algorithm for that user. If found in any one of those clusters, that Web object is fetched from the original server and it is transmitted to the user as well as placed in to the short term cache.

Other Web objects of that cluster will be pre-fetched during browser idle time and will be placed in the short-term cache. If the requested object is not found in any of the clusters of that user, then the required Web page is fetched from the original server and a copy of it is placed in the short term cache as well as transmitted to the requested user. The proposed technique of combining Web caching and pre-fetching makes it possible to increase the hit ratio, decrease the user perceived latency and lower the origin server load.

3.1 Preprocessing step and creation of training dataset

A log file generated by the proxy server consists of time stamp, machine IP size of the requested object, type of method used URL of the requested page, content type etc. While pre-processing, JavaScript files, Cascading style sheet, images that are not requested by the user are removed. A separate log file for individual users for creating inter as well as intra-site Web clustering of Web objects is generated. For creating a training dataset, information is extracted from the traces of log file. Each log file record is converted to the training pattern in the format of

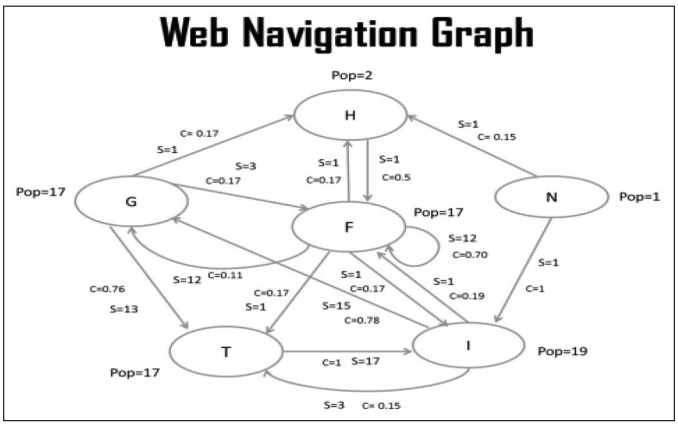

Figure 2: Web Navigation graph for User1

<a1, a2, a3, a4, b> where a1 represents the recency of the Web object, a2 represents frequency of the Web object, a3 represents the size of the Web object, a4 represents the retrieval time (access latency) of the Web object and b represents the class of the Web object. The data type of a1, a2, a3, a4 is numeric and the data type of b is nominal [2].

3.2 Web Navigation Graph (WNG)

A weighted directed Web graph G(x,y) is used to represent the requests of each user, where each node x represents a Web object and each edge y represents a user's transition from one Web object to another. The weight of each edge represents the number of transitions in the set. To make the size of the WNG manageable, the edges are removed whose connectivity between two Web objects is lower than a specified threshold. Support and confidence are the two parameters that determine the connectivity between two objects [4]. Let W:<xi,xj> be an edge from node xi to node xj. Support of G, denoted by freq(xi,xj) is defined as the frequency of navigation steps between xi to xj. Confidence of g is defined as freq(xi,xj)/pop(xi) where pop(xi) is the popularity of xi. By this definition, the support value of the edge (x3,x4) for the user X in Fig. 2 is q(x3,x4) = 1, and confidence value is freq(x3,x4)/pop(x3) = 0.5. If the support threshold chosen is very less, too many less important user's transitions for clustering may be included and if the chosen threshold value is high, many interesting transitions that occur at low levels of support may be missed.

3.3 Web clustering algorithm

The algorithm for clustering inter-site Web pages is described below [4]. A weighted directed Web graph G(x,y) that represents the access patterns of a user is used. This graph is partitioned into sub graphs by filtering edges with low support and confidence values. The nodes in each connected sub graph in the remaining navigational graph will form a cluster. The inputs to this clustering algorithm will be Web navigational graph, the number of users, support threshold and confidence threshold. Remove all the edges with support or confidence value less than the corresponding threshold values. BFS (Breadth First Search) algorithm is applied to the navigational graph. BFS takes a node in the graph (called as source) and visits each node reachable from the source by traversing the edges. It outputs a sub-graph that consists of the nodes reachable from the source. This procedure is applied for all the nodes of the graph. All the nodes in each connected sub-graph forms a cluster. The time complexity of BFS is O(|x|+ |y|) where |x|is the number of nodes and |y|is the number of edges in the graph [4].

google, facebook, google, facebook, google,
hotmail, ndtv, ibnlive, techrunch, Ibnlive,
techrunch, ibnlive, techrunch, ibnlive, google,
facebook, hotmail, facebook, ibnlive, facebook,
google, facebook, google, facebook, google,
hotmail, ndtv, ibnlive, techrunch, Ibnlive,
techrunch, ibnlive, techrunch, ibnlive, google,
facebook, hotmail, facebook, ibnlive, facebook,
google, facebook, google, facebook, google,
hotmail, ndtv ,ibnlive, techrunch, Ibnlive,
techrunch, ibnlive, techrunch, ibnlive, google,
facebook, hotmail, facebook, ibnlive, facebook,
techrunch, ibnlive, google, facebook, hotmail,
facebook, ibnlive, facebook, google, facebook,
google, facebook, google, hotmail, ndtv ,ibnlive,
techrunch, Ibnlive, techrunch, ibnlive,
techrunch, ibnlive, google, facebook, hotmail,
facebook, ibnlive, facebook

Figure 3: User access pattern for User1

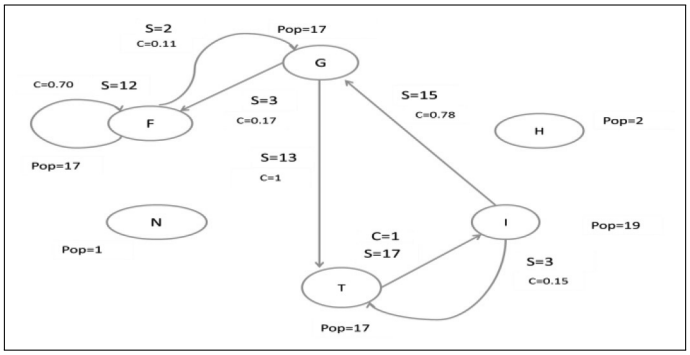

Figure 4: Applying the Support Threshold

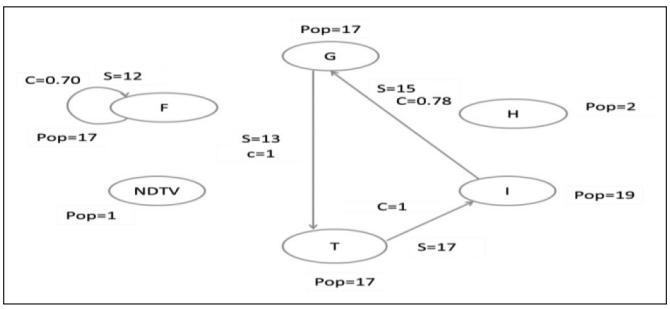

Figure 5: Applying the Confidence threshold

In the above Web Navigation Graph, G stands for Google Web object, H for Hot Mail Web object, N for NDTV Web object, F for Face Book Web object, I for IBN Web object and T for Techrench Web object.

The access pattern for the user1 consists of 6 different Web objects. From the access pattern information, WNG is constructed.

Support and Confidence value for each edge in the WNG is calculated as defined in section 3 and the popularity for each Web object is computed. By assumption Support threshold value is taken as 2 and confidence threshold value as 0.6.

Those edges which have Support and Confidence values less than their threshold are removed. The threshold value chosen for Support and Confidence are critical in specifying the cluster size. It should be noted that the total size of all the objects in a cluster should not exceed the total cache size.

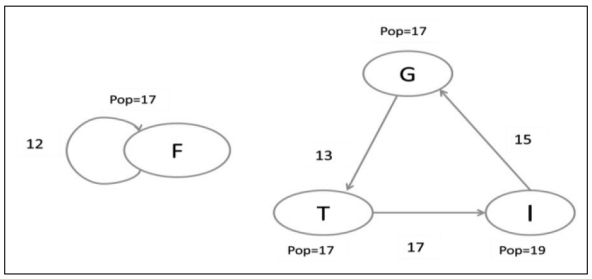

Figure 6: Applying BFS

Then BFS algorithm is applied to the above navigational graph. For each node in the graph known as source, BFS algorithm tries to visit every other node that can be reached from the source by traversing the edges.

It outputs a sub graph that consists of all the nodes reachable from the source. This procedure is iterated until BFS has traversed all the nodes of the initial graph. The nodes in every connected sub graph in the remaining graph forms a Web cluster.

3.4 Pre-fetching using clustering

The following are the steps that take place in the proposed pre-fetching method [4]:

- A user requests a web object. Using the IP address, the proxy identifies the user and maps the user to a particular user group. Given that the clusters of Web objects are known, the proxy searches inside the existing clusters of that user group to find in which cluster the requested object exists.
- All the remaining objects from the selected cluster are pre-fetched from the origin server by the proxy and they are loaded into the short-term cache during the browser idle time.
- The proxy sends to the user his/her requested object.

3.5 Algorithm for the combined intelligent Caching (SVM) and Pre-fetching

Begin
Apply-Clustering algorithm
For each web object m requested by the user
If m is in short-term cache
Begin
Cache hit occurs
Fetch the requested object m from short-term cache
Update the information of m
If the frequency of $m>$ threshold limit
Begin
While no space in the long-term cache for m
Begin
Expel f from long-term cache such that f is in bottom of the long-term cache
End
Class of m=**apply-svm(Common features)**
If class of m=1

```
  Move m to top of the long-term cache
Else
  Move m to the long-term cache
End
End
Else if m is in long-term cache
Begin
  Cache hit occurs
  Fetch the requested object m from the long-term cache
  Update the information of m
  Recalculate the class of m
  If class of m =1
    Move m to the top of the long-term cache
  Else
    Move m to the long –term cache
  End
Else
Begin
  Cache miss occurs
  Fetch m from original server
  Cluster c= getClusterForUser(m)
  If c is not NULL
  Begin
    While no space in short-term cache for web objects in c
    Begin
    Expel object using LRU from short-term cache
    End
  Load the all cluster into the short-term cache during the browser idle time
  End
End
Procedure getClusterForUser(m)
Begin
  For each cluster p for the user
    If m is in cluster p
      p=p+getTheIntrasiteclusters(p);
      return p
  If m is not in any of the cluster
    return NULL
End
End
```

Clustering Algorithm

```
Begin
For each client IP address
Begin
Construct web navigation graph G (U, V)
End
For each G (U, V)
Begin
```

Calculate *Support, Confidence* and *Popularity* of the node U
If *Confidence*< Threshold limit of Confidence
Begin
Remove *V*
End
If *Support*< threshold limit of Support
Begin
Remove *V*
End
Cluster *C* = **Apply BFS** for G (U, V)
End

Breadth first search

Begin
Input: Graph G (U, V)
Choose some starting node *u1*
Mark *u1* as visited
Initialize list *x1* with *u1*
Initialize sub-graph *T* with *u1*
While *x1* is not empty
Begin
Choose node *x2* from front of the list
Visit *x2*
End
For each unmarked neighbor *y*
Begin
Mark *y*
Add *y* to the end of the list *x1*
Add *x2->y* to subgraph *T*
End
Find all neighbors of the node *u1*
Visit each neighbor and mark the visited node
End

Intra clustering Algorithm

Procedure *getTheIntrasiteclusters(m)*
Begin
T=Total number of sessions
For each client IP address
Begin
For each session S
Begin
U=Unique set of Intra-site pages in session S
C=Total number of intra-site pages in session S
For each intra-site page *P* in *U*
Begin
Calculate *Support* and *probability(C/T)* for *P*
If *Support*< threshold limit of Support
Begin
Ignore *P*

Continue;

End

If *Probability*< threshold limit of Probability

Begin

Ignore P

Continue;

End

End

Include P into Cluster C

End End

return C

4 Performance Evaluation

4.1 Dataset

The scheme explained in this paper is tested with a dataset. The dataset is obtained from a proxy server installation ftp://ftp.ircache.net/Traces/DITL- 2007-01-09/. The filename of the dataset used for testing of the scheme is rtp.sanitized-access.20070109.gz under the website. The raw proxy server log file for the dataset contained the details of more than 3 million requests. After the log cleaning process was applied on the dataset, the dataset contained about 610,634 entries fit for analysis. The inner details about the data set are as explained below: (i) Total Number of Items: 610634 (ii) Total Size of Bytes for Dataset: ∼49 GB (iii) Total Number of Items used for *Clustering*: 427443 (iv) Total Size of Bytes Used for *Clustering*: ∼36 GB (v) Total Number of Items used for *Testing*: 183191 (vi) Total Size of Bytes Used for *Testing*: ∼13 GB.

70% of all the requests ordered by time have been used for the user's access pattern analysis, creating training dataset and testing [4].The remaining 30% of the requests were used for testing the scheme.

4.2 Experimental results

Hit Ratio and Byte Hit Ratio Analysis

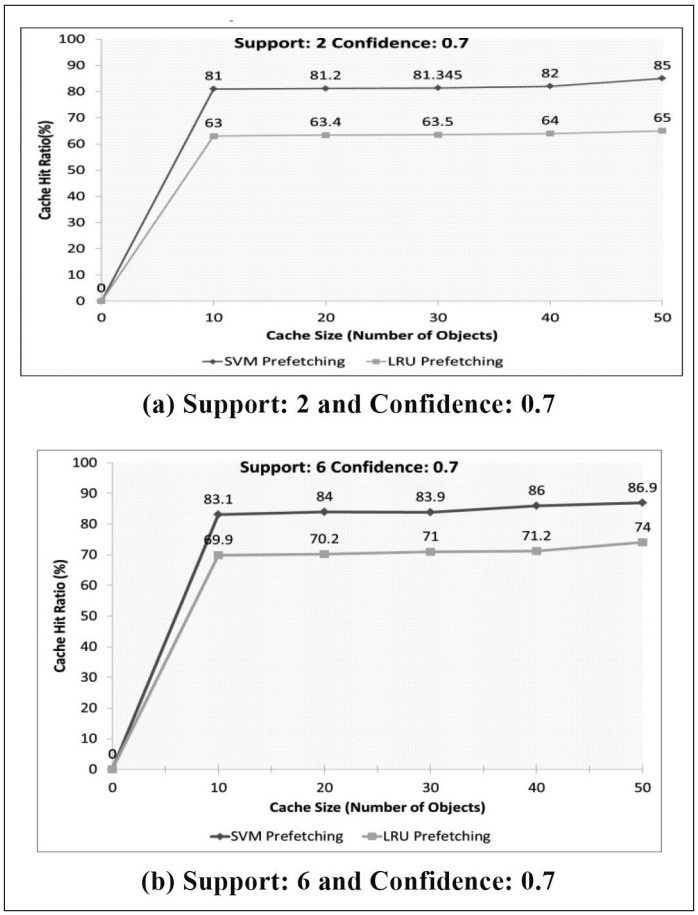

Figure 7: Analysis of HR using SVM and LRU pre-fetching on different values of Support and Confidence

HR and BHR are calculated for different values of Support and Confidence using SVM pre-fetching and LRU pre-fetching. A sample of them is shown above. In the graph, SVM pre-fetching means SVM-LRU caching with pre-fetching and LRU pre-fetching means LRU caching with pre-fetching. Results inferred from the above graphs are stated below:

1. If the Support and Confidence values are increased, it is found that there will be a marginal increase in Hit ratio for increasing cache sizes.

2. Considering Byte Hit Ratio (BHR) it is found that on an average 68% of the total size of the information requested is found fetched from Cache (cache hit) by using SVM-LRU caching with pre-fetching and only 33% of the total size of the requested information is found fetched from the cache using LRU caching with pre-fetching and remaining information are found fetched from the origin server. Considering HR, it is 86% and 67% on average for SVM-LRU and LRU caching with pre-fetching respectively. This shows the superiority of SVM-LRU technique.

   In our earlier paper [3] where intra pages of the requested paper was not considered and LFU technique was used for removal of Web objects from the short-term cache, 64% of

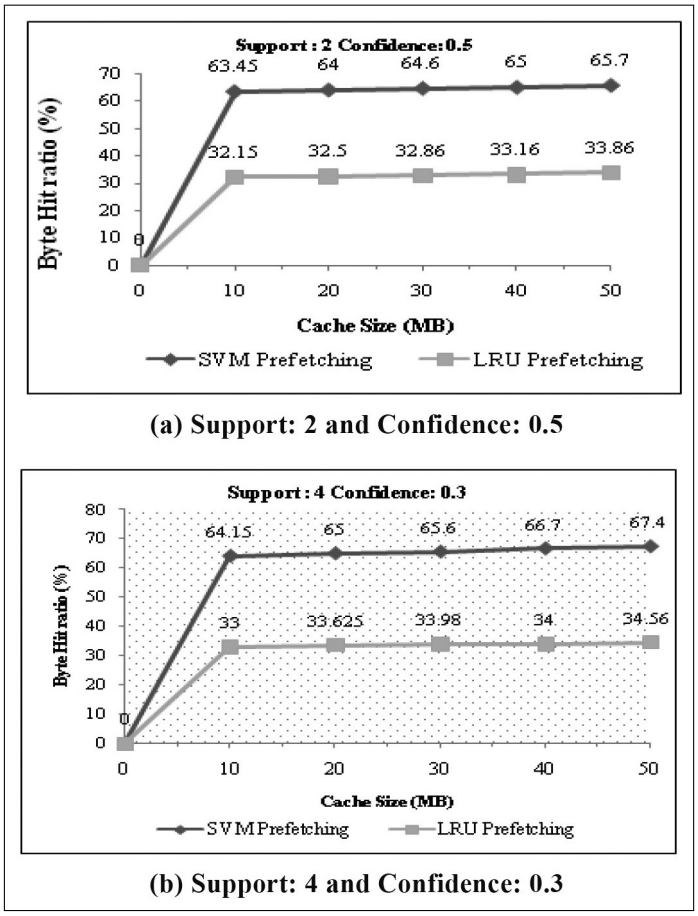

Figure 8: Analysis of BHR using SVM and LRU pre-fetching using different values of Support and Confidence

the total size of the requested information was found fetched from cache due to cache hit (BHR) and 26% of the total size of the information requested was found fetched from cache using LFU pre-fetching and remaining information are found fetched from the origin server. Considering HR, it is 84% and 47% on average for SVM-LFU and LFU caching with pre-fetching respectively.

This shows the superiority of SVM pre-fetching method. By considering the pre-fetching of intra pages of the requested object, the Byte hit ratio BHR is increased by 4% and HR by 2% compared to the Byte hit ratio BHR and HR without considering intra clustering.

3. It is demonstrated that pre-fetching will lead to decrease in network bandwidth utilization and decrease in the access latency because of more cache hits.

5 Conclusion and future work

In this work, a clustering algorithm is used to cluster the Web objects represented in the Web navigation graph. Frequently accessed Web objects are monitored by the Confidence and Support values. If the user's requested Web object is present in the short-term cache then all the other Web objects in that cluster plus all the intra pages of that object are pre-fetched and cached during the browser idle time. If a Web object and its associated pages in the short-term

cache are accessed more number of times than a fixed threshold value then they are moved to long-term cache after classifying them using SVM algorithm. If the requested Web object is found in the long-term cache then all the other Web objects in that cluster are pre-fetched during the browser idle time. If cache miss occurs in both the caches, then that Web object is fetched from the origin server and a copy of it is placed in to the short-term cache. The efficiency of SVM pre-fetching is compared with that of LRU pre-fetching using real data set and it is found that SVM pre-fetching has high HR and high BHR for various values of Support, Confidence and cache sizes. Extension of this work is possible by comparing the efficiency of SVM technique with other machine learning techniques.

Bibliography

[1] Ali W., Shamsuddin S.M., Ismail A.S. (2011), A survey of Web caching and prefetching, *International Journal of Advances in Soft Computing and Its Applications*, 3 (1): 1-27.

[2] Ali W., Shamsuddin S.M., Ismail A.S. (2012), Intelligent Web proxy caching approaches based on machine learning techniques, *Decision Support Systems*, 53(3): 565-579.

[3] Baskaran K.R., Kalaiarasan C., Sasi Nachimuthu A. (2013), Study of combined Web prefetching with Web caching based on machine learning technique, *Journal of Theoretical and Applied Information Technology*, 20th September 2013, 55(2): 280-291.

[4] Pallis G., Vakali A., Pokorny J. (2008), A clustering-based prefetching scheme on a Web cache environment, *Computers and Electrical Engineering*, 34(4): 309-323.

[5] Podlipnig S., Boszormenyi L. (2003); A survey of Web cache replacement strategies, *ACM Computer Surveys*; 35(4):374–98.

[6] Web reference: http://www.wikipedia.com/svm